The PARC 700 Dependency Bank

Tracy Holloway King, Richard Crouch, Stefan Riezler,
Mary Dalrymple and **Ronald M. Kaplan**
Palo Alto Research Center
3333 Coyote Hill Road, 94304 Palo Alto, CA
{thking|crouch|riezler|dalrymple|kaplan}@parc.com

Abstract

In this paper we discuss the construction, features, and current uses of the PARC 700 DEPBANK. The PARC 700 DEPBANK is a dependency bank containing predicate-argument relations and a wide variety of other grammatical features. It was semi-automatically produced and boot-strapped from the output of a deep parser: this allowed for greater consistency of analysis and for more rapid construction.

1 Introduction

The PARC 700 Dependency Bank (DEPBANK) consists of 700 sentences which were randomly extracted from section 23 of the UPenn Wall Street Journal (WSJ) treebank (Marcus et al., 1994). These were parsed by a deep Lexical-Functional Grammar (LFG), converted to the DEPBANK format, and then manually corrected and extended by human validators. Average sentence length is 19.8 words, and the average number of dependencies per sentence is 65.4. The corpus is freely available for research and evaluation; documentation and tools for displaying and pruning structures are also freely available.

The DEPBANK was created because existing treebanks were found inadequate for evaluating predicate-argument structure. In treebanks, this information is usually encoded implicitly in the phrase structure. However, LFG, HPSG, and related grammars encode grammatical functions directly and hence evaluating against tree structures can be difficult. As such, grammar engineers have begun to move away from treebanks to dependency banks (Carroll et al., 2002). In this paper we present PARC's techniques for semi-automatically producing a dependency bank that would be of use for a wide variety of applications, including parser evaluation for a variety of formalisms.

The paper is organized as follows. We first discuss in detail how the DEPBANK was created using both automated and manual techniques (section 2). We then examine several features of the DEPBANK, including tools to specialize it for particular applications (section 3). Finally, we provide some discussion, including current applications for the dependency bank (sections 4 and 6).

2 Constructing the Dependency Bank

This section presents our technique for producing the DEPBANK. We used a combination of automatic and manual techniques in order to get the most accurate results in a reasonable amount of time. The basic process is as follows:

1. Parse the sentence using a broad coverage LFG grammar and bank the functional-structure of the best parse.

2. Convert this automatically to the DEPBANK format, making systematic adjustments.

3. Manually check/correct each structure using pretty-printing and validation tools.

The last step was performed by three linguist validators (two validators per structure).

2.1 Initial Construction

The first step in building the DEPBANK was to parse the 700 sentences with a broad-coverage deep LFG grammar of English using the XLE system (Maxwell III and Kaplan, 1993). For many of the sentences there was more than one possible parse. The best parse was chosen by manual inspection and saved. Note that in a few cases, the best parse was far from the desired parse, as was the case with the sentence in (1) (parc_23.580).

(1) 8 13/16% to 8 11/16% one month; 8 13/16% to 8 11/16% two months; 8 13/16% to 8 11/16% three months; 8 3/4% to 8 5/8% four months; 8 11/16% to 8 9/16% five months; 8 5/8% to 8 1/2% six months.

However, even when the saved parse diverged from the desired one, it was still found to be effective to use it as the basis for the DEPBANK structure rather than create the DEPBANK structure from scratch. This was because subconstituents of the parse were often correct and could be reconnected into the proper structure. For example, in the structure for a sentence like (1), the *month* phrases and the % phrases might be correct and hence only need to be combined into a coordinate structure with the appropriate modifiers.

The second step was to convert the saved grammar f-structure output[1] into the DEPBANK format. This was done automatically and included basic reformatting and some systematic adjustments to the structures. In addition, header information was added to indicate the sentence id number, the validators, etc. The f-structures contain detailed information about grammatical relations. In addition to grammatical functions (e.g., subject, object, adjunct), the f-structures also have information about other syntactically relevant information (e.g., tense, number, adjunct-type). For example, a sentence like *Mary left.* might have the f-structure shown in (2). Note that the f-structures are attribute-value matrices; values of attributes can be either atomic (e.g., *vtype*'s value is *main*) or another f-structure (e.g., *subj*'s value is the f-structure for *Mary*).

[1]The output of the grammar comprises the f-structures and the c(onstituent)-structures. The c-structures are trees. C-structures are not included in the DEPBANK.

(2)
$$\begin{bmatrix} \text{pred} & \text{'leave}\langle\text{subj}\rangle\text{'} \\ \text{subj} & \begin{bmatrix} \text{pred} & \text{'Mary'} \\ \text{ntype} & \text{proper} \\ \text{anim} & + \\ \text{case} & \text{nom} \\ \text{gend-sem} & \text{female} \\ \text{num} & \text{sg} \\ \text{pers} & 3 \end{bmatrix} \\ \text{tns-asp} & \begin{bmatrix} \text{mood} & \text{indicative} \\ \text{perf} & - \\ \text{prog} & - \\ \text{tense} & \text{past} \end{bmatrix} \\ \text{passive} & - \\ \text{stmt-type} & \text{decl} \\ \text{vtype} & \text{main} \end{bmatrix}$$

The conversion process involved a certain amount of "flattening". That is, the highly articulated structure of the grammar output was made less articulate when no loss of information would result. Certain attributes with AVM values were removed, leaving the original AVM value as an attribute of the AVM which contained the eliminated attribute. In particular, attributes which contained no PRED values were flattened. For example, the grammar output in (3a) would be flattened to that in (3b) in which there is no *tns-asp* feature. Note that f-structure *pred* values are assigned an index, e.g., *::0*; this is explained in section 3.1.

(3) a.
$$\begin{bmatrix} \text{pred} & \text{'go}\langle\text{subj}\rangle\text{'} \\ \text{subj} & \begin{bmatrix} \text{pred} & \text{'Mary'} \end{bmatrix} \\ \text{tns-asp} & \begin{bmatrix} \text{tense} & \text{past} \\ \text{mood} & \text{indicative} \\ \text{prog} & + \end{bmatrix} \end{bmatrix}$$

b. subj(go::0, Mary::1),
tense(go::0, past),
mood(go::0, indicative),
prog(go::0, +)

In addition to the flattening, several across-the-board changes were made automatically to the structures to make them more legible and to modify grammatical analyses which were felt to be undesirable. For example, the grammar had *subj*s for all attributive adjectives (e.g., *the flimsy chair*). However, for the purposes of the DEPBANK, it was decided to eliminate these *subj*s. For example, the structure in (4a) for *the red chair* was automatically converted to that in (4b).

(4) a. adjunct(chair::1, red::2),
subj(red::2, chair::1),
adegree(red::2, positive),
atype(red::2, attributive),
adjunct_type(red::2, nominal)

b. adjunct(chair::1, red::2),
adegree(red::2, positive),
atype(red::2, attributive),
adjunct_type(red::2, nominal)

Other examples of this type included the elimination of negative (–) values for *perf* and *prog*, leaving only the positive (+) values. For example, the structure in (5a) for *he left* was automatically converted to that in (5b), while the structure in (6) for *he was leaving* was not changed.

(5) a. tense(leave::0, past),
prog(leave::0, $-$)
b. tense(leave::0, past)

(6) tense(leave::0, past),
prog(leave::0, +)

On a more minor level, several features were renamed to increase legibility, e.g, *attr* became *attributive*; this rewriting was double-checked by the validation tools discussed in the next section.

2.2 Validation

Each DEPBANK structure was manually evaluated by two people. If the structure was not correct, changes were made. In some instances where the grammar effectively did not cover a particular linguistic construction, these changes were substantial, as in (1). In most cases they were minor.

Manual evaluation was made possible by two tools. The first is a pretty printer which displays the DEPBANK structures in a simplified f-structure AVM like structure. This makes the structures more human readable without altering the application-friendly DEPBANK structure. Consider the DEPBANK structure for *Don't jump yet.* (parc_23.313); this is shown in (7a) and its pretty-printed equivalent is in (7b). There is a second pretty-print format (not shown) which displays the index numbers as part of the AVM format.

(7) a. structure(
mood(jump::0, imperative),
adjunct(jump::0, not::5),
adjunct(jump::0, yet::4),
stmt_type(jump::0, imperative),
subj(jump::0, pro::1),
vtype(jump::0, main),
pers(pro::1, 2),
pron_type(pro::1, null),
adegree(yet::4, positive),
adv_type(yet::4, sadv),
adjunct_type(not::5, negative))

b.
```
| pred        'jump'
| mood        imperative
| stmt_type   imperative
| vtype       main
| subj        | pred           'pro'
              | pers           2
              | pron_type      null
| adjunct     | pred           'yet'
              | adegree        positive
              | adv_type       sadv
| adjunct     | pred           'not'
              | adjunct_type   negative
```

The second tool checked for valid structures. This validation tool performs two types of tasks. The first is to determine whether a structure is notationally well-formed. For example, it checks whether header information is formatted correctly, whether commas, parentheses, and colons are correctly placed, and whether the structure is fully connected. These checks eliminate common typos that occur when making changes to the structures.

The second task of the validation tool is to check for user-defined substative well-formedness requirements. For example, all the possible feature names are declared (e.g., *subj*, *num*, *tense*), as are their possible values (e.g., *adegree* can have the values *comparative*, *positive*, *superlative*). If a feature is found that was not predeclared or was associated with an incorrect value, then the tool reports it to the validator. This helps locate typos and is useful for finding naming inconsistencies.

In addition to listing possible feature names, the validators could list co-occurrence and other well-formedness conditions. For example, *xcomp*s (e.g. infinitives, small clauses) must have *subj*s; *mod*s (i.e. nouns in noun-noun compounds) must have *pers* and *num* but not *atype*; *conj*s (i.e. coordinate structures) must have *coord_level* and *coord_form* but not *subj*s; pronouns with a *pron_type* of *expletive* must not have a *pro* value.

These checks were invaluable in two circumstances. The first was when the original banked structure was incorrect and had to be modified by the validator. This, for example, occurred relatively frequently where the wrong choice had been made between a noun and an adjective modifier; this distinction is important because it has semantic reprecussions, as witnessed by the frequent lexicalization of noun-noun compounds (e.g., *tractor trailer*). The word *red* in a phrase like *the red box* has the structure in (8a) when it is an adjective and the structure in (8b) when it is a noun.

(8) a. adjunct(box::1, red::2),
adegree(red::2, positive),
atype(red::2, attributive)
adjunct_type(red::2, nominal)

b. mod(box::1, red::2),
num(red::2, sg),
pers(red::2, 3)

Since the change from adjective to noun and vice versa required several concurrent changes, the validating tool can check that all the relevant changes have been made.

The second circumstance where this validation tool was used was where the grammar had been incomplete. For example, it was discovered that not all conjunctions provided a *coord_form* (in particular, lexical conjunctions such as *and* and *or* had a *coord_form*, but punctuation conjunctions such as *;* and *:* did not). To provide consistency in the DEPBANK, these missing forms were added, and the validation tool was used to bring the absence of the feature to the validator's attention. Note that many across-the-board changes were made by the automatic converter from the output of the grammar to the DEPBANK structures, e.g., the elimination of *subj*s in attributive adjectives (section 2.1). This decreased the changes the validators had to make and hence the chance for human error.

In conclusion, although the process of creating the DEPBANK was labor intensive, the extremely detailed results were made possible by (1) using a deep grammar to bootstrap the initial structures and (2) having validation tools to double check for wellformedness at the level of typos and of grammatical structure.

3 Features of the Dependency Bank

In this section, we discuss the contents of the DEPBANK structures themselves. Much of this information can be found in greater detail in the on-line documentation. The choice of format and of the dependencies is extremely important since it dictates what applications the DEPBANK can be used for. First we discuss indices, reentrancies, and stemming (section 3.1). We then discuss the dependencies chosen (section 3.2) and finally mention problems with redundant information in the DEPBANK (section 3.3).

3.1 Indices, Reentrancies, and Stemming

All predicates in a given DEPBANK structure are assigned a unique index, with the matrix predicate always being assigned the index *::0*. One reason for the indices is to distinguish two instances of the same word. For example, in *Most estimates for Monsanto run between $1.7 and $2 a share.* (parc_23.328), there are two distinct instances of the predicate *$:* one of them is designated *$::10* and one *$::11*. These in turn have their own modifiers (i.e., *1.7* for *$::10* and *2* for *$::11*).

The second use of the indices is for reentrant structures, i.e., structures in which a single item is related to more than one predicate. This occurs with controlled infinitives, with small clauses, and with the second argument of copular constructions. Consider *Of course, the health of the economy will be threatened if the market continues to dive this week.* (parc_23.315). The noun phrase *the market* is the subject of *continues* and of *dive*. This is shown in (9).

(9) subj(continue::8, market::11),
xcomp(continue::8, dive::6),
subj(dive::6, market::11)

There are two important points about the analysis in (9). The first is that the *subj* of the infinitive is indicated even though it does not appear in the string in canonical subject position, i.e., immediately before *to dive*. The second is that the fact that the two subjects are the same is indicated by their identical index *::11*.

The example in (9) also demonstrates the stemming used in the DEPBANK. The surface form *continues* is stemmed to *continue*. Note that the features for *market*, shown in (10), indicate that it is third person singular, and hence is compatible with the third singular surface form of the verb.

(10) num(market::11, sg),
pers(market::11, 3)

Stemming and indices provide a uniform, easy to manage format for indicating predicates. More difficult is deciding which dependencies to have in the DEPBANK; this is dicussed next.

3.2 Dependencies Chosen

The most difficult decisions in creating the DEPBANK involved deciding which dependencies and features to keep. There were two main types of dependencies at issue: dependencies representing surface information and redundant dependencies. As will be discussed in more detail below, the general approach was to keep redundant information because it is easier to delete it using the DEPBANK structure-pruning tool than to go back and add it.

All argument and adjunct relations are indicated in the DEPBANK structures — without this the structures would not be properly connected. The possible grammatical functions include (see (Butt et al., 1999)):

(11) a. Subcategorized functions: *subj*, *obj*, *obj_theta* (secondary objects), *comp* ('that'- and 'whether'-clauses), *xcomp* (infinitives, small clauses, and postcopular arguments), *obl* and *obj_ag* and *obl_compar* (subcategorized obliques)
b. Nonsubcategorized functions: *adjunct*, *name_mod* (used in person names), *mod*, *topic_rel* and *pron_rel* (used in relative clauses), *focus_int* and *pron_int* (used in interrogatives), *poss*, *conj*, *number*, *quant* and *aquant* (quantifiers)

Information of this type is included in all dependency banks, although it is only indirectly present in most tree banks. In general, determining these grammatical relations was not difficult, other than the *obl-adjunct* distinction, and the output of the grammar was quite reliable.

However, the DEPBANK includes a large number of syntactic features in addition to those indicating the role of the phrase in the clause. These include information about: statement type; tense, mood, aspect, and passivization; person, number, gender, and case; determiners; comparatives and superlatives; adjunct and adverb types. These features can be extremely useful for certain applications and as such they are included in the DEPBANK. An example full structure was shown in (7). For applications which need only predicate-argument structure, a structure pruning tool is provided which allows the user to specify which features to keep. The tool will not prune the core grammatical function features. That is, if the value of a feature is something with an index, it cannot be pruned. For example, *num* can be pruned because its values are *sg* and *pl*, but *subj* cannot be pruned because its value is indexed and the resulting structure would be disconnected. In addition, it is possible to keep features only when they have certain values. For example, someone might wish to keep *adegree* only when it has the values *comparative* or *superlative* but not *positive*.

There are two classes of features that are present in the grammar output, but are eliminated in the DEPBANK structures. The first class are features that exist solely for grammar internal reasons, e.g., to test certain well-formedness constraints. Fortunately, these are all contained within the feature *check* and so deleting a single feature *check* as part of the automatic conversion program eliminates all these features (section 2.1). The second class involves features that are inconsistently present in the grammar output and were not hand-corrected by the validators because they were not felt to be important for the DEPBANK. For example, people's names, e.g., *Mary*, provide an *anim* + feature in the grammar. However, not all animates have this feature in the grammar output, e.g., *girl* does not. Because of this, the *anim* feature is felt to be misleading since its absence does not indicate an inanimate. So, *anim* is eliminated from the DEPBANK structures.

As a final note, there are certain other types of information that are not included in the DEPBANK. One of these is word sense disambiguation: only the stemmed form of the word is indicated, without any information as to its sense in the clause. In addition, word order is not indicated other than in the recording of the original string in the *sentence_form* field of each DEPBANK structure. For example, in *I don't feel either hard or soft*. (parc_23.384), there is no indication in the DEPBANK structure as to which of the adjectives is first in the coordination. Instead, they are both *conj*s of *coord::6*,[2] as seen in (12).

(12) conj(coord::6, soft::18),
conj(coord::6, hard::17),
coord_form(coord::6, or),
coord_level(coord::6, AP),
precoord_form(coord::6, either)

As we have seen, the DEPBANK includes extremely detailed grammatical information, well beyond the level of predicate argument structure. As discussed in the next section, this level of de-

[2]*coord* is a special predicate introduced where the grammar output has the set structure used for coordination in LFG. It represents the elements of the f-structure set by having one *conj* attribute for each set member, i.e. for each conjunct.

tail, while useful for many applications, can provide difficulties for others.

3.3 Double Counting Information

Some of the information in the DEPBANK is redundant for certain applications. That is, if the output of an application is matched against a DEPBANK structure, it may be the case that if it matches *feature1* then it will always also match *feature2*, and that if it misses *feature1* then it will always also miss *feature2*. This is undesirable for some evaluation measures and training scenarios since the result is a double credit or a double penalty.

For example, imperative constructions are indicated in two ways. They are assigned *stmt_type imperative* and *mood imperative*. In addition, the vast majority of imperatives have a null pronominal *subj*. This was seen above in *Don't jump yet.* (parc_23.313); the relevant dependencies are repeated in (13).

```
(13) stmt_type(jump::0, imperative),
     mood(jump::0, imperative),
     subj(jump::0, pro::1),
     pers(pro::1, 2),
     pron_type(pro::1, null)
```

So, whenever the *stmt_type imperative* is found, the *mood* and *subj* information follow automatically. However, for non-imperative constructions, the *subj* value is not predictable, and the *mood* information cannot be derived from the *stmt_type* and vice versa, e.g., *stmt_type declarative* can correspond to *mood indicative* or *subjunctive*.

Another example comes from pronouns, which in the DEPBANK are assigned the indexed value *pro* except for expletive *it* and *there*. The form of the pronoun provides a wealth of information about the pronoun. For example, in parc_23.374 the pronoun *she* has the features in (14).

```
(14) case(pro::1, nom),
     gend_sem(pro::1, female),
     num(pro::1, sg),
     pers(pro::1, 3),
     pron_form(pro::1, she),
     pron_type(pro::1, pers)
```

The case, gender, number, person, and pronoun type information can all be derived by knowing that the *pron_form* was *she*. For applications where this is a problem, everything but the *pron_form* could be eliminated.[3] However, there are situations where being able to search for all instances of, for example, personal pronouns is necessary and so having this information overtly recorded via the *pron_type* feature is useful.

[3]The pronouns are stemmed, but their case can be recovered from their grammatical function

At this time, a complete list of "doubled" dependencies is not available. However, as they are found, the structure pruning tool can eliminate them if they interfere with a given application. By having this pruning tool available, the possible applications of the DEPBANK are increased.

Thus, as the detailed syntactic information found in the DEPBANK is not necessary or even desirable for all applications, a structure pruning tool is provided to eliminate unwanted dependencies from the DEPBANK.

4 Applications

We used the DEPBANK to evaluate a stochastic parsing system consisting of an LFG grammar and a stochastic disambiguation model trained on sections 02-21 of the UPenn WSJ treebank (Riezler et al., 2002; Crouch et al., 2002).

In an evaluation of a combined system of parser and stochastic disambiguator, three types of parse selection are compared against the gold standard of the DEPBANK: (i) *lower bound*: random choice of a parse from the set of analyses (averaged over 10 runs), (ii) *upper bound*: selection of the parse with the best F-score, and (iii) *stochastic*: the parse selected by the stochastic disambiguator.

Table 1: F-score results for parser evaluation against DEPBANK.

lower bd.	stochastic	upper bd.	error red.
76.6	79.5	85.2	34

Table 1 shows F-scores for these parse selections, where F-score is defined as $2\times$ *precision* $\times$ *recall* / (*precision* + *recall*). The disambiguation accuracy of the stochastic selection system is assessed in a window defined by a random choice and the best possible choice from the parses delivered by the symbolic parsing system. The reduction in error rate relative to these upper and lower bounds achieved by the stochastic disambiguation system is noted in the *error red.* column of Table 1.

In addition to an overall figure, precision, recall, and F-score results can be broken down according to separate dependencies. Fig. 1 shows a breakout of evaluation scores according to the dependencies in the parses selected by the stochastic disambiguation system.

We have also used the DEPBANK in grammar development. In order to have an accurate assessment of grammar coverage, it is not sufficient to know simply whether a sentence parses or not. By comparing the grammar output against the dependency structure, it is possible to determine whether the grammar produced the correct structure, e.g., is the correct noun phrase the subject? Independent of a stochastic disambiguation system, it is possible for the grammar writer to determine how close the match is by inspecting evaluation scores computed for the upper bound selection. The result is not all-or-nothing, but rather a gradient score for each dependency or for the overall matching result.

DEPBANK tools have also been used for rebanking other parse banks, where the best (though not necessarily correct) analysis has been handchosen. Grammar changes require updates to the parse bank. This is achieved semi-automatically by chosing the new parse with the closest matching dependency structure to the original analysis.

5 Comparisons to Other Banks

The DEPBANK is closest in form and intent to the gold-standard dependency annotations proposed by (Carroll et al., 1999) for 500 sentences selected from the Brown corpus. As discussed in more detail in (Crouch et al., 2002), the DEPBANK (a) eliminates residual aspects of surface structure that intruded into the Carroll *et al* annotations, and (b) provides a more detailed range of grammatical features. In addition, it provides a genuinely random selection of sentences (i.e. not preselected to be parsable by a particular parser/grammar), taken from the *de facto* standard corpus for parser evaluation. An evaluation of a WSJ-trained stochastic parsing system for LFG achieved 76% F-score on Carroll et al.'s test set (see Riezler et al. (2002)).

The Tiger treebank (Brants et al., 2002) for German newspaper texts is on a much larger scale (some 30,000 sentences) than the PARC 700 DEPBANK, and also makes use of the XLE as an annotation tool, but is quite different in form and intent.

DEPENDENCY	PREC.	REC.	F-SCORE
number-type	96	95	96
coord-form	92	93	93
ptype	92	91	91
psem	89	92	90
pron-form	88	89	89
number	91	88	89
vtype	91	88	89
inf-form	97	80	88
det-type	88	86	87
perf	90	85	87
pers	87	87	87
num	86	87	86
tense	86	86	86
deixis	80	89	84
quant	85	80	83
case	83	80	82
passive	80	83	82
stmt-type	80	79	80
prog	94	69	79
proper	79	79	79
pron-type	74	82	78
atype	80	75	77
adegree	81	72	76
obj	75	75	75
poss	74	76	75
adjunct-type	74	74	74
xcomp	74	73	74
subj	73	73	73
obl	64	83	72
comp	78	64	70
obl-ag	74	65	69
prt-form	72	65	68
precoord-form	100	50	67
adjunct	67	63	65
conj	68	62	65
gerund	53	80	64
adv-type	71	57	63
mod	59	62	60
pcase	60	56	58
topic-rel	48	72	57
coord-level	56	56	56
focus-int	49	63	55
partitive	40	83	54
comp-form	32	54	40
obj-theta	42	36	39
topic	20	67	31
obl-compar	29	29	29
OVERALL	80.0	78.8	79.5

Figure 1: Evaluation scores broken down according to (selected) dependency relations.

It is a *treebank* supplemented with some grammatical relations annotations, rather than a dependency bank. By abstracting away from details of surface structure, the DEPBANK provides a more transparent and articulated account of predicate-argument/adjunct and other semantically relevant structures. The intent of Tiger is to provide a resource for both the training and evaluation of language processing; the DEPBANK is targeted at the evaluation of systems performing deeper language analysis. The Prague Dependency Bank (Hajič, 1998) for Czech is constructed in a similar fashion as the Tiger treebank, using POS tagging and a basic grammar to bootstrap the manual treebank annotation. Unlike the Tiger treebank, the Prague Dependency Bank does not encode linear order, although it does use tree structures to encode dependencies. Unlike the DEPBANK, it primarily encodes relations between words, i.e., predicate-argument structure, and not other syntactic dependencies, although some of these, such as number and tense, can be derived from the POS tags.

The LinGO-Redwoods treebank (Oepen et al., 2002) supplements its trees with explicit semantic representations, and is thus also aimed at providing resources for deeper language analysis. However, the LinGO-Redwoods treebank does not provide a gold-standard annotation in that it only records the *best* analysis produced by a particular grammar and parser, which need not necessarily be the *correct* analysis. It is also very heavily tied to the HPSG formalism, whereas the DEPBANK attempts to generalize away from its LFG heritage.

6 Conclusion

The PARC 700 DEPBANK is a dependency bank containing both predicate-argument relations and a wide variety of other grammatical features. It was semi-automatically produced and boot-strapped from the output of a deep parser. Without this automation and boot-strapping, consistency of analysis would have been extremely difficult to achieve, and the time involved would have been exorbitant. Although the process is not entirely automated, it is hoped that the system can be used to efficiently create further dependency banks.

The DEPBANK has so far been applied to evaluate the combined system of parser and stochastic disambiguator presented in Riezler et al. (2002). Since most state-of-the-art statistical parsing systems are based on the UPenn WSJ treebank, the DEPBANK can directly be applied for a dependency-based evaluation of such parsing systems. Furthermore, due to the possibility of pruning unwanted dependencies, the PARC 700 DEPBANK may be useful to evaluate also parsing systems that were not trained or created from the UPenn treebank. Moreover, the grammar development possibilities opened up by the DEPBANK can be exploited for parsers that were developed for corpora other than newspaper text like the WSJ.

References

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*, Sozopol.

Miriam Butt, Tracy Holloway King, Maria-Eugenia Niño, and Frédérique Segond. 1999. *A Grammar Writer's Cookbook*. CSLI Publications.

John Carroll, Guido Minnen, and Ted Briscoe. 1999. Corpus annotation for parser evaluation. In *Proceedings of the EACL workshop on Linguistically Interpreted Corpora (LINC)*. Bergen, Norway.

John Carroll, Anette Frank, Dekang Lin, Detlef Prescher, and Hans Uszkoreit, editors. 2002. *Proceedings of the Workshop on Parseval and Beyond and the 3rd International Conference on Language Resources and Evaluation (LREC'02)*. ELRA.

Richard Crouch, Ronald M. Kaplan, Tracy Holloway King, and Stefan Riezler. 2002. A comparison of evaluation metrics for a broad coverage stochastic parser. In *Proceedings of the Workshop on Parseval and Beyond and the 3rd International Conference on Language Resources and Evaluation (LREC'02)*. Las Palmas, Spain.

Jan Hajič. 1998. Building a syntactically annotated corpus: The prague dependency treebank. In *Issues of Valency and Meaning*, pages 106–132. Karolinum, Prague.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn treebank: Annotating predicate argument structure. In *ARPA Human Language Technology Workshop*.

John T. Maxwell III and Ronald M. Kaplan. 1993. The interface between phrasal and functional constraints. *Computational Linguistics*, 19:571–589.

Stephan Oepen, Ezra Callahan, Dan Flickinger, Christopher Manning, and Kristina Toutanova. 2002. LinGO redwoods: a rich and dynamic treebank for HPSG. In *Proceedings of the Workshop on Parseval and Beyond and the 3rd International Conference on Language Resources and Evaluation (LREC'02)*. Las Palmas, Spain.

Stefan Riezler, Tracy Holloway King, Ronald M. Kaplan, Richard Crouch, John T. Maxwell III, and Mark Johnson. 2002. Parsing the Wall Street Journal using a Lexical-Functional Grammar and discriminative estimation techniques. In *Proceedings of hte 40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*. Philadelphia, PA.